CPU Design

In this project, you will design, simulate, and synthesize your own 16-bit CPU. Please complete all the steps below:

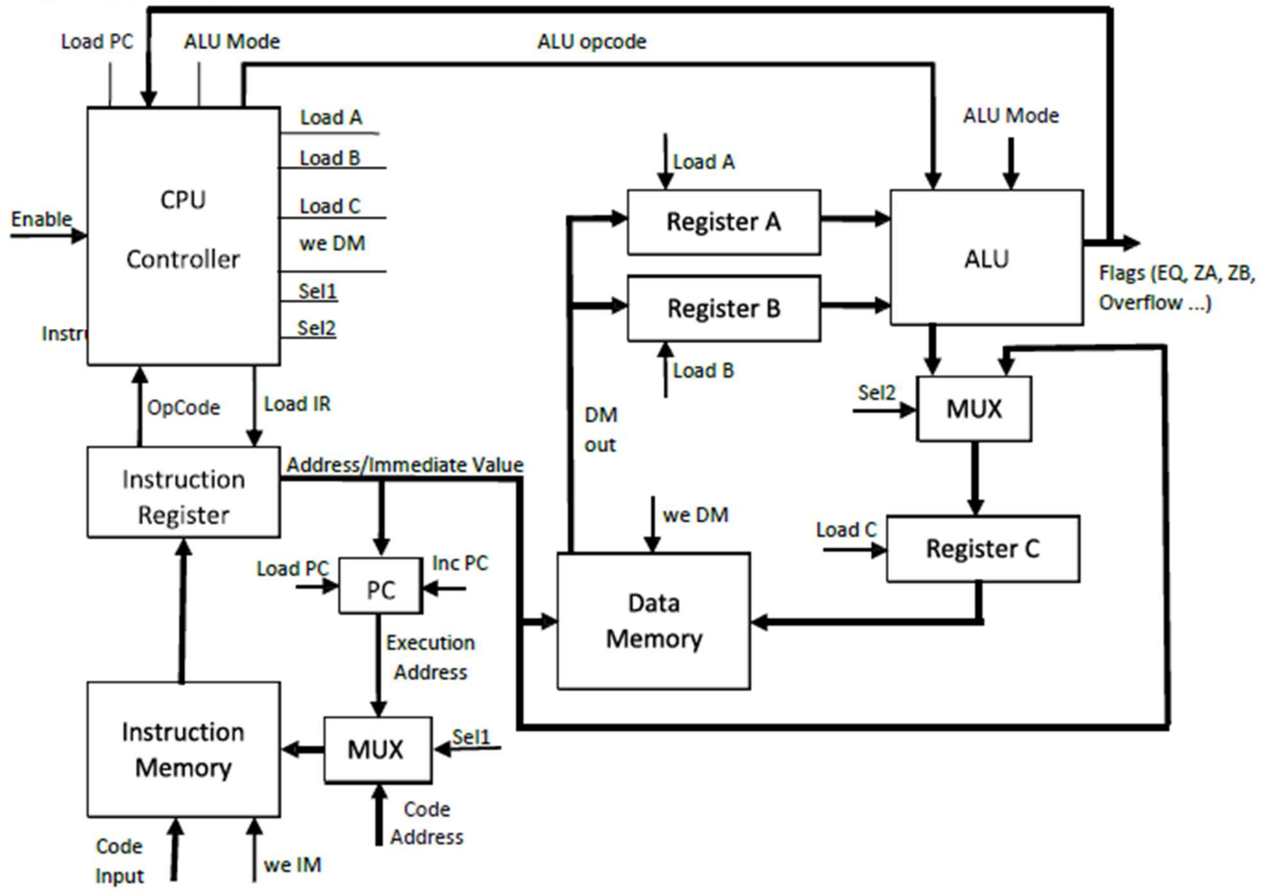
1. Design your instruction formats, mnemonics, and machine codes per the following requirements.

   a) Operations:

   - Store instructions into your Instruction Memory
   - LDA: Load a value from DM into Register A
   - LDB: Load a value from DM into Register B
   - LIC: Load an immediate value entry into Register C
   - STC: Store the value of Register C back to the Data Memory
   - RDM: Read the data stored in Data Memory at a specific address
   - NOP: Do nothing
   - HLT: It halts the execution.
   - Jumps: Add any three conditional jumps using the flags from the ALU.
   - ALU ops: Output of ALU operations (using Project 5) are stored in Register C.
   - Bonus: *Revise ALU to include "division" and "subtraction" operations.*
   - *Please note that your design must include instruction format, instruction set, and mnemonics per the general guidelines:*

Opcode (LIC)	*Data*
Opcode (LDA, LDB, STC)	*Address*
Opcode (ALU ops)	*N/A*
Opcode (Jumps)	*Address*

b) Architecture:



CPU Design

A central processing unit (CPU) is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions.
Most modern CPUs are microprocessors, meaning they are contained on a single integrated circuit (IC) chip. An IC that contains a CPU may also contain memory, peripheral interfaces, and other components of a computer; such integrated devices are variously called microcontrollers or systems on a chip (SoC).

- **Controller:** The control unit of the CPU contains circuitry that uses electrical signals to direct the entire computer system to carry out stored program instructions. The control unit does not execute program instructions; rather, it directs other parts of the system to do so. The control unit communicates with both the ALU and memory.

- **Arithmetic logic unit:** The arithmetic logic unit (ALU) is a digital circuit within the processor that performs integer arithmetic and bitwise logic operations. The inputs to the ALU are the data words to be operated on (called operands), status information from previous operations, and a code from the control unit indicating which operation to perform. Depending on the instruction being executed, the operands may come from internal CPU registers or external memory, or they may be constants generated by the ALU itself. When all input signals have settled and propagated through the ALU circuitry, the result of the performed operation appears at the ALU's outputs. The result consists of both a data word, which may be stored in a register or memory, and status information that is typically stored in a special, internal CPU register reserved for this purpose.

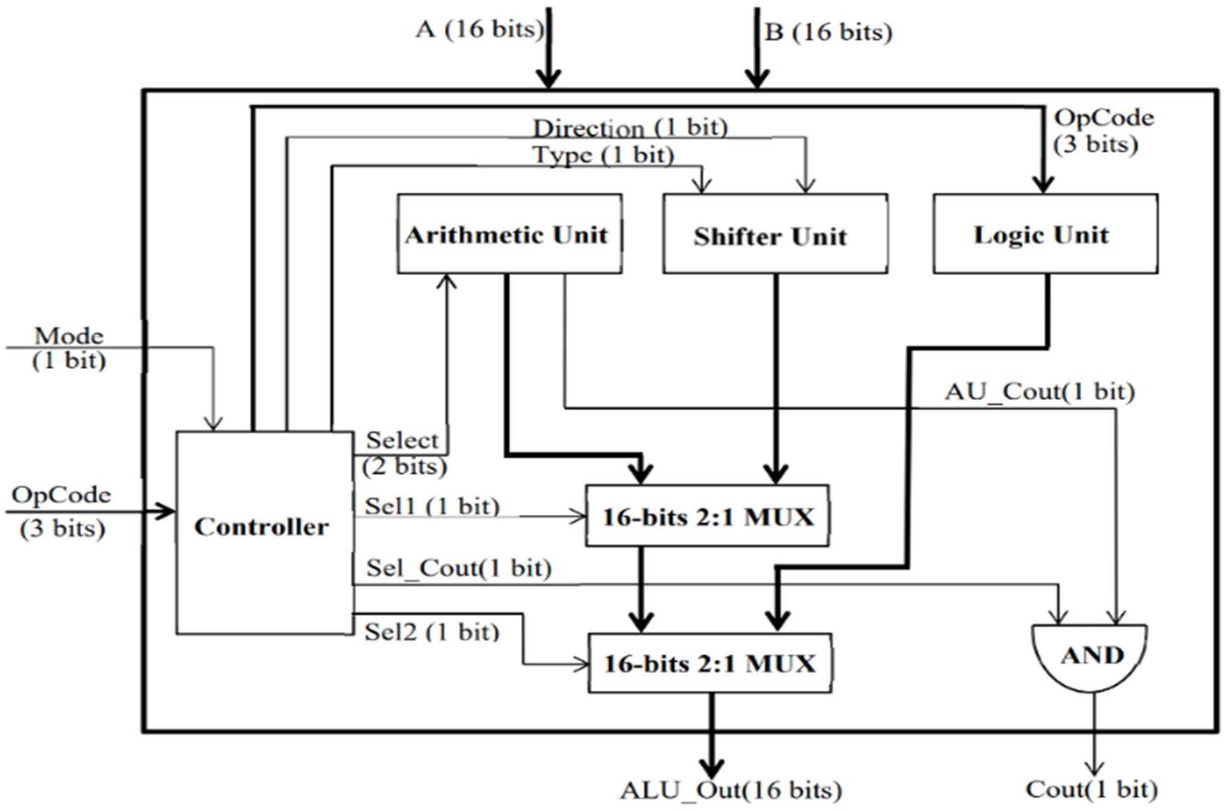


Figure 1: ALU Diagram

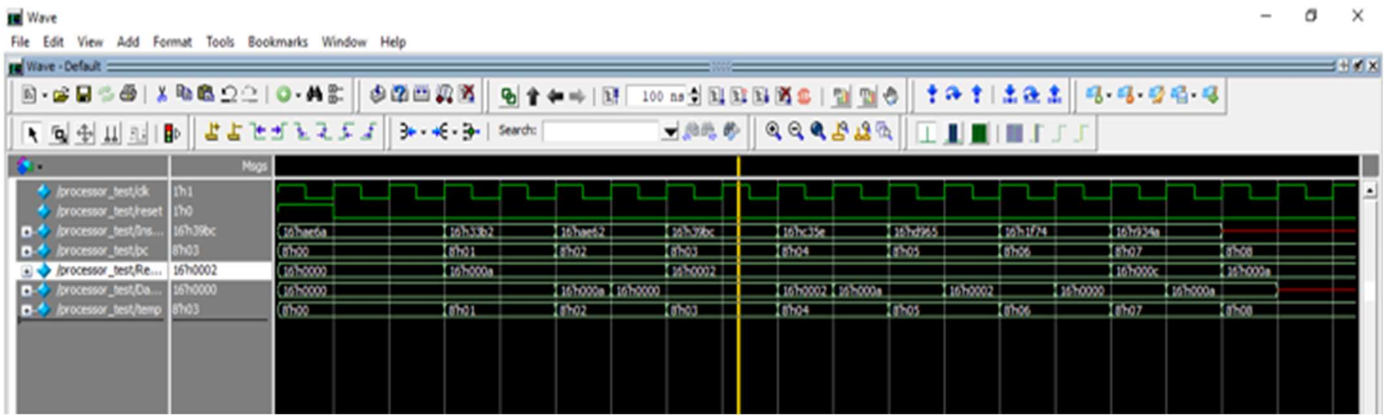
- **Instruction Memory, Instruction Register and Program Counter:** Instruction Memory holds all the instructions, their opcodes and immediate data or addresses as required. This is loaded into the Instruction Memory during compilation. Program Counter keeps the count of the address on which the program is currently executing and points to the address of next instruction to be executed.
  Based on the address provided by the Program Counter, the Instruction Memory outputs one Instruction with relevant address/data which is stored into the Instruction Register for execution.

3. Your testbench must include a "good" set of instructions that would test the functionality of your CPU. They MUST NOT be HARDCODED into your memory. They should be stored into your CPU using your testbench *either as a machine code or assembly code.*

   Here is an introductory sample assembly code. *Please note that you can use different mnemonics and instructions sets.*

   - LDA "0011" ; Register A = DM(0011)
   - LDB "0110" ; Register B = DM(0110)
   - ADD ; Register C = Register A + Register B
   - STC "0001" ; DM(0001) = Register C
   - AND ; Register C = Register A AND Register B
   - STC "0010" ; DM(0010) = Register C
   - RDM "0001" ; DM Out = DM(0001)
   - LDB "0011" ; Register B = DM(0011)
   - HLT ; The execution shall HALT.
   - RDM "0001" ; Should not execute.

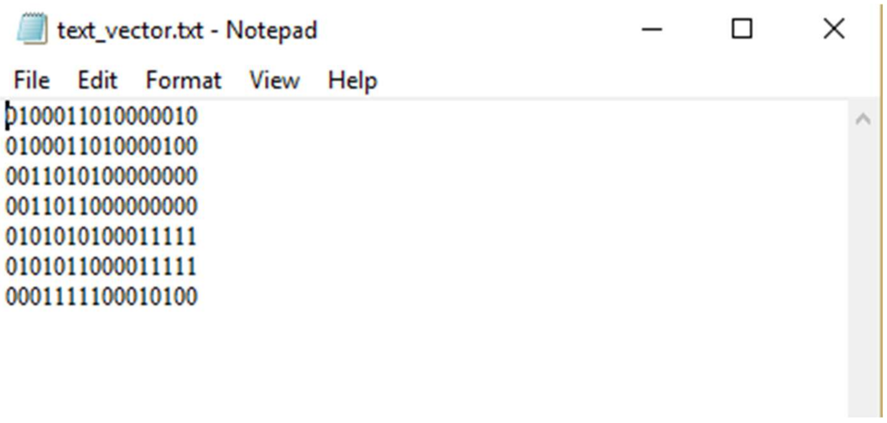
Waveform:


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
100 ns
Search:
Msgs
/processor_test/clk 1'h1
/processor_test/reset 1'h0
/processor_test/Ins... 16'h39bc
/processor_test/pc 8'h03
/processor_test/Re... 16'h0002
/processor_test/Da... 16'h0000
/processor_test/temp 8'h03
16'hae6a 16'h33b2 16'hae62 16'h39bc 16'hc35e 16'hd965 16'h1f74 16'h934a
8'h00 8'h01 8'h02 8'h03 8'h04 8'h05 8'h06 8'h07 8'h08
16'h0000 16'h000a 16'h0002 16'h000c 16'h000a
16'h0000 16'h000a 16'h0000 16'h0002 16'h000a 16'h0002 16'h0000 16'h000a
8'h00 8'h01 8'h02 8'h03 8'h04 8'h05 8'h06 8'h07 8'h08

In the first clock cycle, the instruction is loaded into IR and in second cycle, the controller generates control signal and we get output at the end of second cycle.

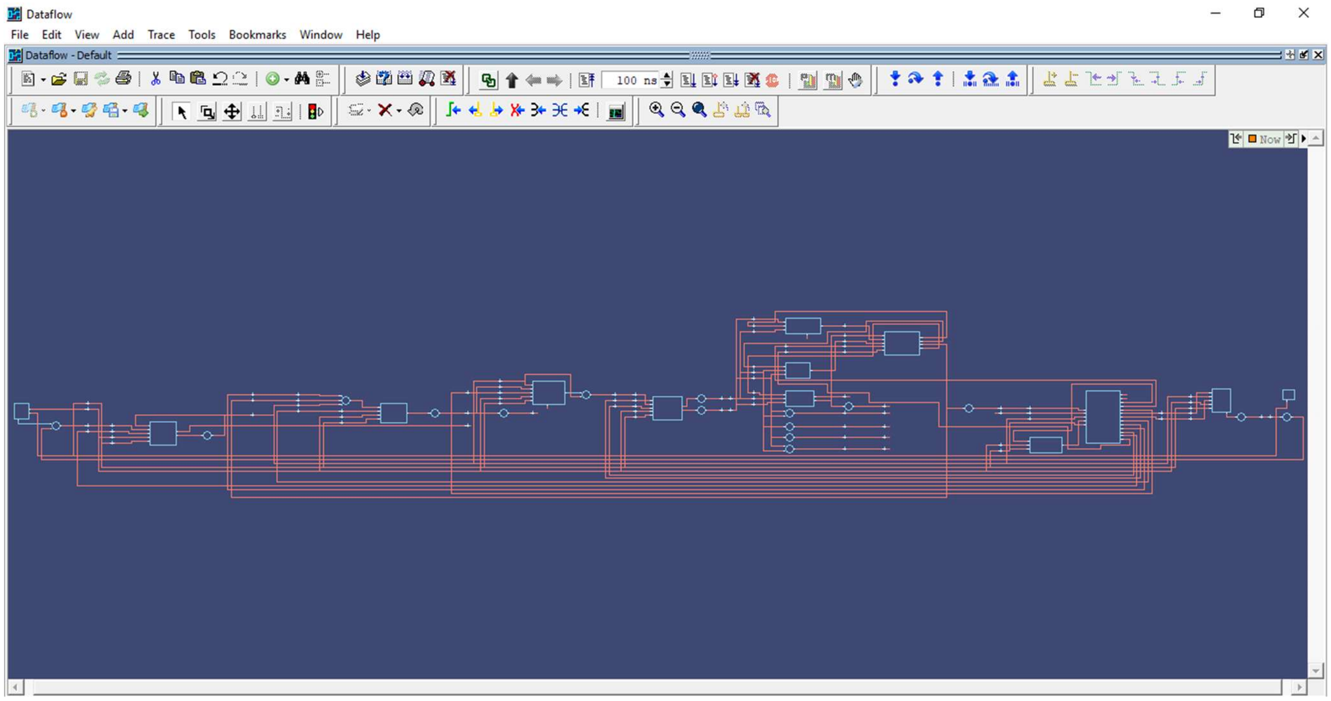
CPU Design

Test Vectors:



```
0100011010000010
0100011010000100
0011010100000000
0011011000000000
0101010100011111
0101011000011111
0001111100010100
```

Flow Diagram:


Dataflow
File Edit View Add Trace Tools Bookmarks Window Help
Dataflow - Default
100 ns
Now